Preprint
UCRL-JC-142262

On the Topology of the Level Sets of Scalar Field

V. Pascucci

This article was submitted to
Association of Computing Machinery Symposium on Computational Geometry 2001, Medford, MA, June 3-5, 2001

December 12, 2000

U.S. Department of Energy

Lawrence
Livermore
National
Laboratory

DISCLAIMER

This work was performed under the auspices of the United States Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

Center for Applied Scientific Computing - Lawrence Livermore National Laboratory

On the Topology of the Level Sets of a Scalar Field.

Valerio Pascucci

Abstract

This paper introduces a new simple algorithm for the construction of the Contour Tree of a 3D scalar field augmented with the Betti numbers of each contour component. The algorithm has $O(n \log n)$ time complexity and $O(n)$ auxiliary storage, where n is the number of vertices in the domain of the field. The algorithm can be applied to fields of any dimension in which case it computes the Contour Tree augmented only with the Euler characteristic of each contour. The complexity in any dimension remains $O(n \log n)$. This is the same complexity as in [4] but with correct computation of the tree for fields with bounded domains.

1 Introduction

Scalar fields are used to represent data in different application areas like geographic information systems, medical imaging or scientific visualization.

One fundamental visualization technique for scalar fields is the display several isocontours: that is sets of points of equal scalar value. For example in terrain models isolines are used to highlight regions of equal elevation. In medical CT scans an isosurface can be used to show and reconstruct the separation between bones and soft tissues.

In molecular modeling the simplex representation of the interaction energy between two molecules is a scalar field defined over a six dimensional configuration space. The six dimensions are the three translational and the three rotational degrees of freedom of the relative positions of the two molecules. The isocontours of the field represent all the configurations energetically equivalent.

The domain of the scalar field is typically a geometric mesh and the scalar field is provided by associating each vertex in the mesh with a sampled scalar value. If the mesh is a simplicial complex then a piecewise linear field is naturally defined by interpolating linearly, within each simplex, the scalar values at the vertices.

The Contour Tree is a graph that represents the relations between the connected components of the isocontours in a scalar field. Two connected components that merge together (as one changes continuously the isovalue w) are represented as a two arcs that join in a node of the graph. The pre-computation of the Contour Tree allows to collect structural information relative to the isocontours of the field. This can be used for example to speedup the computation of the isosurface [12]. The display [2] of the Contour Tree provides the user with direct insight on the structure of the field. Extensions beyond the 3D case [4] are particularly useful because for higher dimensional data [1] the traditional rendering schemes are less intuitive. Especially in those cases the complementary display of the Contour Tree would enhance the user interaction time necessary to "understand" the structure of the data.

One fundamental limitation of the Contour Tree is the lack of additional information regarding the topology of the contours. In high pressure chemical simulations [10], our application of reference, hydrogen bonds between the atoms cannot be represented in a tradition way,

but can be characterized by isosurfaces of potential fields. The Contour Tree provides important information regarding the clustering of atoms into molecules but fails to discriminate between linear chains and closed rings (or more complex structures) that have different physical behaviors. For example the approach in [8] computes the Betti numbers of each isosurface extracted to characterize explicitly its topology. Computationally such an approach becomes very expensive since it requires the explicit computation of the isosurface itself. If applied as a preprocessing stage to precompute all the Betti numbers of all the isocontours in a scalar field, it would have $O(n^2)$ complexity.

The first efficient technique for Contour Tree computation in 2D was introduced by de Berg and van Kreveld in [5]. The algorithm proposed has $O(n \log n)$ complexity. A simplified version with same complexity in 2D and $O(n^2)$ complexity in higher dimension was proposed by van Kreveld et al. in [12]. This new approach is also used as a preprocessing for an optimal isocontouring algorithm. It computes a small seed set from which any contour can be tracked in optimal running time. In the extended version of this paper [12] we also prove a lower bound of $O(n \log n)$ for the computation of the Contour Tree (by reduction to sorting) which shows the optimality of the $O(n \log n)$ complexity. The approach has been improved by Tarasov and Vyalyi [11] achieving $O(n \log n)$ complexity in the 3D case by a three pass mechanism that allows to resolve the different types of criticalities. Recently Carr, Snoeyink and Axen [4] presented and elegant extension to the general dimensional case based on a two pass scheme that builds a merge-tree and a split-tree that are successively composed into a unique Contour Tree. The approach achieves optimal $O(n \log n)$ time complexity in any dimension.

This paper introduces a new algorithm for the construction in 3D of an augmented Contour Tree where each arc of the tree is associated with the Betti numbers of each contour. The first Betti number β_0 of a mesh, in this case the isosurface, is the number of connected components of the mesh. The second number β_1 is the number of tunnels of the mesh. The third β_2 is the number of voids enclosed in the mesh. With the Betti numbers one provides a complete topological characterization of all the isosurfaces. The algorithm provides this additional topological information within the optimal $O(n \log n)$ computation time. The extension to higher dimensions only computes the Contour Tree (not augmented with the Betti numbers) maintaining the same $O(n \log n)$ complexity. This is the same complexity as in [4] but in a more robust framework. The presence of "special cases" like multiple vertices with the same function value or open isocontours due to a bounded domain are handled uniformly. In particular note that the solution proposed in [4] to add a layer of constant values around the mesh has the undesired effect of arbitrarily merging different components of the isocontour as shown in Figure 1. This modifies the Contour Tree making it unsuitable for example for the computation of seed sets (we plan to use it in combination with the approach proposed in [3]). This modification of the Contour Tree can provide partially misleading information to the user if displayed in an graphical interface.

2 The Contour Tree

Consider a scalar field $\mathcal{F}$ defined as a pair (f, M), where f is a real valued function and M is the domain of f. In the following the domain mesh M is assumed to be a simplicial complex with n vertices $\{v_1, \ldots, n_n\}$. The function f is restricted within each simplex $\sigma \in M$ to be the linear interpolation of the values at the vertices. In other words the field $\mathcal{F}$ is completely defined by the mesh M and the set of scalar values $\{f_1, \ldots, f_n\}$ where $f_i = f(v_i)$. Since M is connected (or processed one connected component at a time) the range of f is a simple

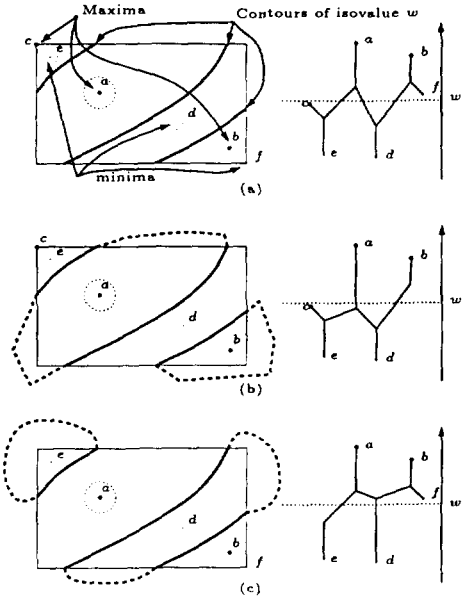

Figure 1: (a) A 2D scalar field with its Contour Tree on the right. The contours of the level set of isovalue w are displayed in thick solid lines. (b-c) Two possible modifications of the scalar field and Contour Tree due to the insertion of an outer layer of uniform values.

closed interval $r = [f_{min}, f_{max}]$ with $f_{min} = \min\{f_1, \ldots, f_n\}$ and $f_{max} = \max\{f_1, \ldots, f_n\}$.

For simplicity of presentation M is also assumed to be a d-manifold with boundary (∂M is a (d-1)-manifold) homeomorphic to a d-ball. This assumption holds in most practical cases and technically can be removed (making the following discussion more complicated).

One fundamental way to study the field $\mathcal{F}$ is to extract its level sets. For a given scalar w the level set $L(w)$ of isovalue w is defined as the inverse image of w onto M through f:

$$L(w) \doteq f^{-1}(w).$$

We call each connected component of the level set $L(w)$ a *contour*. One aspect that is well understood in Morse theory [9] is the evolution of the homotopy types of the contours of $\mathcal{F}$ while w changes continuously in r. The points at which the topology of a contour changes are called critical points and are assumed to be isolated. This assumption is not satisfied in general but can be enforced by a small perturbation of the function values $\{f_1, \ldots, f_n\}$. This perturbation procedure is here weakened by simply assuming that the $\{f_1, \ldots, f_n\}$ are sorted from the smallest to the largest so that $i < j \Rightarrow f_i \leq f_j$. This can be enforced with an $O(n \log n)$ preprocessing. In the following the order of the f_i is used to resolve non-isolated criticalities using the concept of filtration as in [6].

We follow the notation of [4] and define the Contour Tree as a graph $\mathcal{T}$ whose vertices are associated each with a function value f_i and whose connectivity represents the relation among the contours of $\mathcal{F}$ as follows.

- Each leaf of $\mathcal{T}$represents a local extremum where a contour is created or destroyed. for continuous changes of w. The function value of the extremum is associated with the leaf nodes of $\mathcal{T}$.
- Each interior vertex of $\mathcal{T}$represents the merging and/or splitting of two or more contours for continuous changes of w. The function value at which a split/merge occurs is associated with the node.
- Each arc of $\mathcal{T}$ represents a contour that remains isolated for w ranging between the function values associated with the extremes of the arc.

Figure 2 shows a 2D scalar field with the associated Contour Tree. Note that the Contour Tree is not a complete Morse graph of $\mathcal{F}$ since the topological changes of a single contour are not recorded. A more intuitive way to characterize the Contour Tree is the following informal definition:

Definition 1 *The Contour Tree of $\mathcal{F}$ is the graph obtained by contracting each contour of $\mathcal{F}$ to a single point.*

In the following we show how the Contour Tree can be efficiently computed and augmented in 3D with the complete topological

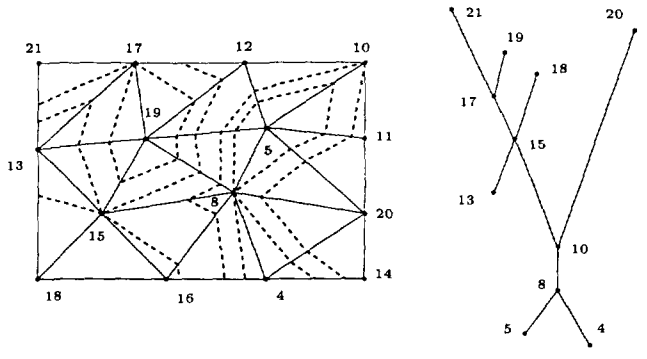

Figure 2: (a) 2D triangular mesh with the contours of the saddles. (b) Corresponding contour tree.

information of each contour. In particular we mark each arc of the tree with the Betti numbers of the corresponding component specifying if it is open or closed and the number of its tunnels. This implies that additional nodes are added to $\mathcal{T}$ dividing each arc at the critical points where the topology of the corresponding contour changes. For example the event of a contour changing its topology from a sphere to a torus would not be noted in the basic Contour Tree but is registered in this augmented Contour Tree.

3 Construction of the Contour Tree

This section discusses the algorithm for the construction of the augmented Contour Tree. We first give an outline of the approach, and provide the details in the following subsection.

3.1 Algorithm Outline

The algorithm used to build the augmented Contour Tree is based on two successive sweeps of the data (after sorting the vertices by function value).

Sweep 1. The mesh is processed from maximum function value to minimum function value. In this stage a set of "merge simplices" are marked.

Sweep 2. The mesh is processed from minimum function value to maximum function value. In this stage the Contour Tree is constructed and the Betti numbers of all the contours are reported.

The idea is to build a filtration of the domain mesh M such that processing the simplices in the filtration order is equivalent to sweeping the level sets of $\mathcal{F}$. Once this goal is achieved one can define the set of simplices where contours merge and apply the original Contour Tree construction scheme in [12] maintaining optimal $O(n \log n)$ running time in any dimension. Because the second sweep actually tracks the topology of the contours one is also able to determine their Betti numbers without penalty on the running time complexity.

3.2 Conceptual Mesh Decomposition

This section introduces a convenient way to decompose the domain mesh into a set of convex cells. Such a decomposition does not need to be computed explicitly but represents a good way to reason about the sweep procedure that is performed for increasing/decreasing values of w.

Consider the sorted set of function values $\{f_1, \ldots, f_n\}$ defined at the vertices of the mesh. We know that:

$$f_i \leq f_{i+1} \qquad i = 1, \ldots n-1.$$

Each edge $\overline{v_i, v_j}$ of the mesh is divided at its midpoint $v_{i,j}$. Consider a k-simplex $\sigma \in M$. The global ordering of the vertices induces on the vertices of the simplex the order $(v_{i_0}, \ldots, v_{i_k})$. σ is partitioned into $k+1$ convex cells by slicing with k hyperplanes (k-1-flats) as follows. For each $j \in \{0, \ldots, k-1\}$ partition the vertices of σ into two sets $V' = \{v_{i_0}, \ldots, v_{i_j}\}$

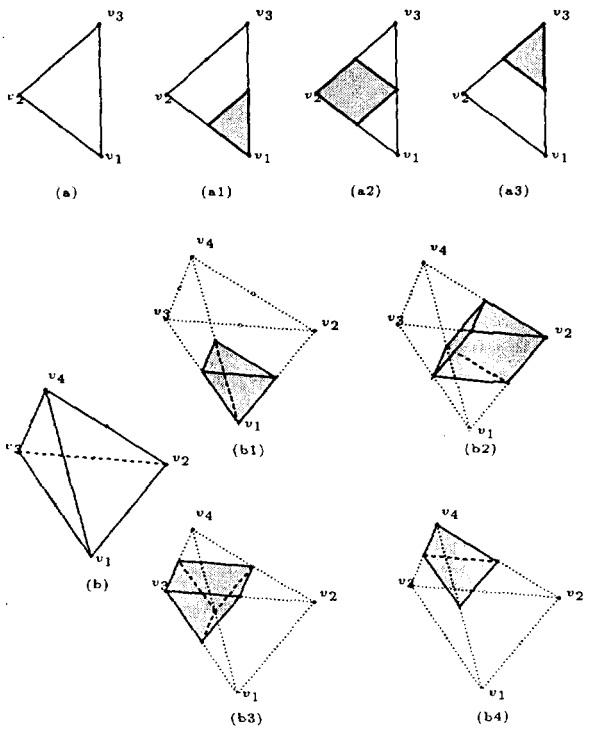

Figure 3: (a) Decomposition a triangle into three convex cells (a1-a3). (b) Decomposition of a tetrahedron into four convex cells (b1-b4).

and $V'' = \{v_{i_{j+1}}, \ldots, v_{i_k}\}$. It is easy to see that the set of midpoints $v_{h,k}$ with $v_h \in V'$ and $v_k \in V''$ span a (k-1)-flat. This (k-1)-flat is used to divide the simplex. Figure 3(a) shows the decomposition of a triangle into three convex cells show in gray in (a1),(a2) and (a3). The same decomposition procedure for a tetrahedron is shown Figure 3(b), where the resulting convex gray regions are (b1),(b2),(b3) and (b4).

Since each simplex has been decomposed into a constant number of pieces this new mesh M' has size $O(n)$. The mesh M' has the following additional property:

Theorem 1 *For any isovalue* $w \in r\backslash\{f_1, \ldots, f_n\}$ *there is a subcomplex* M^w *of* M' *that is homeomorphic to the level set* $L(w)$ *of* $\mathcal{F}$.

Proof: Since $w \notin \{f_1, \ldots, f_n\}$ one can partition the vertices of the original mesh in the set V_g of those with function value greater than w and the set V_l of those with function value smaller than w. The set V_c is the set of vertices of M' that split an edge of M having one extreme in V_l and one extreme in V_g. The complex M^w is the set of cells in M' that have all the vertices in V_c and is homeomorphic to $f^{-1}(w)$. In fact if the vertices of M^w are moved to the actual interpolation points along the edges of M then M^w is deformed to become coincident with $f^{-1}(w)$ without any structural modification. ◇

The order of the vertices of the mesh M' is derived from the order of the vertices of M. In particular the original vertices of M maintain the same relative order but are interleaved with the new ones. Consider a vertex v_i and the set of all the vertices $v_{j_1}, \ldots, v_{j_k}$ connected to v_i with an edge of M and such that $i < j_1 < \ldots < j_k$. The ordered vertices of M' will contain the subsequence $\{v_i, v_{i,j_1}, \ldots, v_{i,j_k}, v_{i+1}\}$. That is v_i is immediately followed by the vertices introduced on the edges incident both to v_i and to a vertex of index greater than i. Notice that this is a unique canonical order derived from the order of the vertices of M.

3.3 Critical Points? No Thanks

The main characteristic of the approach proposed here is the construction of the Contour Tree performed without resolving explicitly the criticalities of the field. The classification of the critical points is replaced with the filtration technique used in [6]. That is the characterization of a point as critical is derived from its effect on the topological structure of the contours as w crosses its function value.

Definition 2 *A* filtration *is a sequence of simplicial complexes where every complex is a proper subcomplex of its predecessor.*

Given the subdivided mesh M' defined in the previous section we define a filtration by ordering all the cells in M' in a sequence. Note that

in the following it is not necessary for M' to be a simplicial complex, but one can reduce to that case by applying a canonical decomposition to all its convex cells. The sequence $\mathcal{S}$ of cells in M' is called the *filter* of the filtration. Each prefix of $\mathcal{S}$ is a complex in the filtration.

We use the order of the Vertices of M', as defined at the end of the previous subsection, to define the filter as follows. The filter $\mathcal{S}$ is built incrementally in dimension by inserting successively the simplices of increasing dimension. Initially $\mathcal{S}$ contains only the sequence of vertices of M'.

Step A. In general consider the cells of dimension k. The k-cells are sorted by considering the indices of their faces already in $\mathcal{S}$. The cell a precedes the cell b if the largest index of a facet of a is smaller than the largest index of a facet in b. Ties are broken by looking at the second, third, etc. largest face index in the cell (two k-cells must differ for at least one face).

Step B. After the sorting the k-cells are grouped by equal largest index of their faces. If the (k-1)-cell a already in the filter $\mathcal{S}$ is immediately followed by b then the group of k-cells that have a as face of largest index is inserted in $\mathcal{S}$ between a and b.

Steps A and B are repeated d times for $k = 1, 2, \ldots, d$ until all the cells of M' are in $\mathcal{S}$. If $\mathcal{S}$ has m elements we call $\mathcal{S}_i$, with $i = 1, \ldots, m$ the prefix of its first elements. M'_i is the complex containing all the simplices in $\mathcal{S}_i$. The sequence of the M'_i, with $i = 1, \ldots, m$ is the desired filtration.

Theorem 2 *For any isovalue $w \in r\backslash\{f_1, \ldots, f_n\}$ there is an integer i such that the complex $\partial M'_i$ contains every contour in $L(w)$.*

Proof: Call j the index in $\mathcal{S}$ of the first vertex with function value greater than w. We choose $i = j - 1$. It easy to see that ∂M_i contains the subcomplex of M' used in the proof of Theorem 1. In particular the difference between $\partial\mathcal{S}_i$ and $L(w)$ is an eventual set of boundary faces of M'. In other words $L(w) = \partial M_i \backslash \partial M'$. ◇

Processing the filtration is equivalent to sweeping the data with the level set $L(w)$, with w that changes continuously from f_{min} to f_{max}. The main advantage is that the filtration automatically splits the events in simple split merge operations. Note that we do not need to identify eventual multiple/degenerate criticalities since they are resolved incrementally by the order of the simplices in the filter. Moreover since we are looking at contours of non-critical values of w we have no particular problem in dealing with the boundary effects.

Proposition 1 *Any contour C of isovalue $w \in r\backslash\{f_1, \ldots, f_n\}$ is a (d-1)-manifold possibly with boundary.*

Proposition 2 *Any contour C of isovalue $w \in r\backslash\{f_1, \ldots, f_n\}$ partitions M into two connected regions.*

Call $f^+(w)$ the region of M where the field f is greater than w:

$$f^+(w) = f^{-1}((w, f_{max}]).$$

Proposition 3 *Consider the effect of increasing continuously the isovalue w from w_a to w_b, with $w_a, w_b \in r\backslash\{f_1, \ldots, f_n\}$. If two contours merge together then for decreasing isovalue w from w_b to w_a there are two connected components of $f^+(w)$ that merge together.*

This property applied to the filtering justifies the equivalence between the first pass of the Contour Tree computation algorithm [4] and the first pass of the Betti number computation algorithm [6]. The only difference is that the latter identifies also the facet at which two contours merge together.

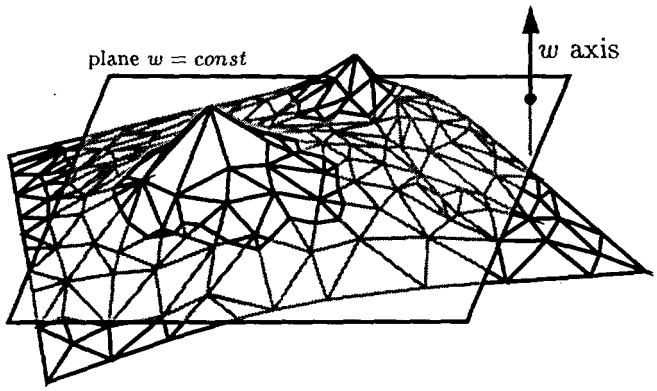

Figure 4: Sweep of the mesh M with a hyperplane $w = const$

3.4 Algorithm Details

As mentioned in the outline, the algorithm is based on two sweep stages. Each sweep is equivalent to processing the filter $\mathcal{S}$ in direct order or in reverse order.

Sweep 1. First of all we note that the decomposition M', and hence the filter $\mathcal{S}$, does not need to be constructed explicitly. The data is instead swept maintaining only the level set $L(w)$ while w is changed continuously as shown in Figure 4. The vertices of each level set are approximated by the midpoints of the edges of M. Each step in the sweep operation consists in moving one vertex of M from "below" $L(w)$ to "above" $L(w)$. The vertices are moved in the order defined initially with the sorting. It is easy to see that this sweep operation is equivalent to the incremental processing of $\mathcal{S}$ as long as instead of computing the contours at the critical points we simply add the new simplices accordingly to the order of the vertices in M'.

In this way we can perform a first sweep from f_{max} to f_{min} as in [6] by traversing the adjacency graph (dual graph) of the d-simplices of M we determine (like in a union-find) what volumes merge. We then mark the facets where the actual merge occurs. Since we do not build M' explicitly we mark one vertex in the facet since it corresponds to one facet in M'.

Sweep 2. The second sweep is performed in the same way but from f_{min} to f_{max}. In this sweep we determine with a direct test what components we are merging (like in a union-find) by attributing one name to each component and verifying if the new simplex we insert connects two simplices with different name. From the information collected in the previous sweep we know what contours split. Since we know also at what cell the actual split occurs we can perform the tandem search as in [12] to determine efficiently what components will results after the split. In particular as the facet c marked in the first sweep is inserted in the current $L(w)$ then we have two contours that are separated everywhere other than the shared facet c. Starting from c we can traverse the two contours doing one step on each of them alternately. As soon as the the smaller contour is traversed completely the tandem traversal stops and the cells of the small contour are traversed again to be assigned a new name. The fact that in each split operation we perform work proportional to the smallest contour guarantees that the amortized cost of the split operations is globally $O(n \log n)$.

3.5 Tracking the Betti Numbers

Since we traverse and maintain the level sets $L(w)$ we also have the opportunity to count the number of the simplices of any dimension that they contain. In particular in 3D we count the number of vertices, edges and triangles in each contour. Moreover we count the number of boundary edges.

Each contour is by definition one connected component of $L(w)$, so its first Betti number β_0 is 1.

The last Betti number β_2 is 0 if the number of boundary edges is greater than 0 (one open surface). Otherwise the contour is a closed surface with $\beta_2 = 1$.

The second Betti number β_1 is be computed as in [8] from the linear equation given by the

Euler number of the surface:

$$\sum_{i=0}^{d}(-1)^i\beta_i = \sum_{i=0}^{d}(-1)^i\nu_i$$

where ν_i is the number of i-cells in $L(w)$.

During the second sweep we determine from this equation the current vale of β_1 so that each contour is completely characterized topologically. Notice that this allows to determine indirectly what are the additional critical points of the field that do not cause the merge or split of any contour.

4 Non-manifold Complex Domains

One can deal with domains having non-simple topology and non-manifold features. If the topology of the domain is not as simple as a d-ball then two contours can split for increasing w even if no two components of $f^+(w)$ merge for decreasing w (see for example Figure 5). In this case Proposition 3 is not true. Moreover non-manifold structures of the domain make it possible for two contours to merge/split at a simplex that is not a (d-1)-face of M'. To deal with these cases one has to modify the first sweep stage such that in the incremental processing of the filter $\mathcal{S}$ the actual contours (or unions of them) are maintained to determine the merge simplices. Still one can mark one simplex as the simple connection between two contours so that it can be used as the starting element of the tandem search.

5 Betti Numbers at Critical Points

Until now we have characterized all the contours but those that are part of a level set $L(w)$ with $w = f_i$ and $i \in \{1,\ldots,n\}$. Even if $\{f_1,\ldots,f_n\}$ is a set of measure 0, one may still

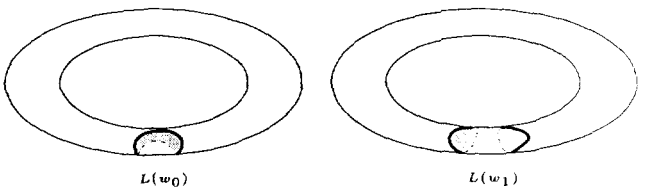

Figure 5: Level sets computed for a toroidal domain. One component of the level set $L(w_0)$ that splits into two components at $L(w_1)$ without merging of two regions of $f^+(w)$ from $w = w_1$ to $w = w_0$.

be interested in knowing the topological characterization of one contour for exactly $w = f_i$ for $i \in \{1,\ldots,n\}$. This needs some additional computation if f_i is the function value at a critical point. In particular we use the following fact.

Proposition 4 *The level set $L(w)$ is a* deformation retract *of the region $f^{-1}([w-\epsilon, w+\epsilon])$ if w is the only critical value in the interval $[w-\epsilon, w+\epsilon]$.*

In other words $L(w)$ has the same homotopy type of $f^{-1}([w-\epsilon, w+\epsilon])$. Consequently we can compute the Betti numbers of the latter in place of the former. This is easily accomplished using a subset of the filter $\mathcal{S}$.

Call j_1 the index in $\mathcal{S}$ of the first vertex of M with function value equal to f_i, and j_2 the index in $\mathcal{S}$ of the first vertex of M with function value greater than f_i. We use the sub-filter $\mathcal{S}^* = \mathcal{S}_{j_2-1} \backslash \mathcal{S}_{j_1-1}$ to build a filtration. The first complex in the filtration is the subcomplex of M' that is homeomorphic to $L(w-\epsilon)$. This is the complex that we normally track during the second sweep of the Contour Tree construction algorithm. The following complexes in the filtration are built by adding the cells of $\mathcal{S}^*$ one by one. $\mathcal{S}^*$ is traversed twice. The first traversal allows to determine which cell is going to merge with which contour of $L(w-\epsilon)$. The second traversal is performed one component at a time and is equivalent to the second pass of the Betti number computation algorithm [6]. In this way

we determine the Betti numbers of each component of $f^{-1}([w-\epsilon, w+\epsilon])$. The initialization of the Betti numbers is done using the Betti numbers of each component of $L(w-\epsilon)$ as computed before.

6 Conclusions

In this paper we have introduced a new efficient algorithm for the pre-computation of all the Betti numbers of all the contours in a scalar field. The algorithm generates the Contour Tree structure augmented with the combinatorial characterization of each contour. The running time is $O(n \log n)$ which is optimal for the computation of the Contour Tree alone.

The key novelty of the scheme is the use of the concept of filtration for tracking the topological changes of the contours in a mesh. This will allow to apply the thoery of topological persistence [7] to scalar fields. As a consequence we plan to obtain a sound definition of a hierarchical Contour Tree to use as a basis for a multiresolution representation of the corresponding scalar field.

7 Acknowledgments

Special thanks go to Ken Joy and Herbert Edelsbrunner for the interesting discussions that helped developing this work. Thanks also go to Peter Lindstrom for his useful suggestion that helped to improve the presentation of this paper.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract no. W-7405-Eng-48.

References

[1] C. Bajaj, V. Pascucci, G. Rabbiolo, and D. Schikore. Hypervolume visualization: a challenge in simplicity (color plate p. 172). In *Proceedings of the 1998 Symposium on Volume Visualization (VOLVIS-98)*, pages 95–102, New York, October 19–20 1998. ACM Press.

[2] Chandrajit L. Bajaj, Valerio Pascucci, and Daniel R. Schikore. The contour spectrum. In Roni Yagel and Hans Hagen, editors, *IEEE Visualization 97*, pages 167–175. IEEE, November 1997.

[3] Praveen Bhaniramka, Rephael Wenger, and Roger Crawfis. Isosurfacing in higher dimensions. In *IEEE Visualization 2000*, pages 267–273, Salt Lake City, Utah, October 2000.

[4] Hamish Carr, Jack Snoeyink, and Ulrike Axen. Computing contour trees in all dimensions. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 918–926, January 2000.

[5] Mark de Berg and Marc J. van Kreveld. Trekking in the alps without freezing or getting tired. *Algorithmica*, 18(3):306–323. July 1997.

[6] Cecil Jose A. Delfinado and Herbert Edelsbrunner. An incremental algorithm for betti numbers of simplicial complexes on the 3-sphere. *Computer Aided Geometric Design*, 12(7):771–784, 1995. ISSN 0167-8396.

[7] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *Proceeding of The 41st Annual Symposium on Foundations of Computer Science*. IEEE, November 2000.

[8] Shirley F. Konkle, Patrick J. Moran, Bernd Hamann, and Kenneth I. Joy. Fast methods for computing isosurface topology with

betti numbers. Submitted to Proceedings of Dagstuhl 2000, F. Post, G. Neilson and G.-P. Bonneau, eds.

[9] J. Milnor. *Morse Theory*, volume 51 of *Annals of Mathematics Studies*. Princeton University Press, 1963.

[10] Eric Schwegler, Giulia Galli, and Francois Gygi. Water under pressure. *Physical Review Letters*, 84(11):2429–2432, 2000.

[11] Sergey P. Tarasov and Michael N. Vyalyi. Construction of contour trees in 3d in o(n log n) steps. In *Proceedings of the fourteenth annual symposium on Computational geometry*, pages 68–75, Minneapolis, June 1998. ACM.

[12] M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. In *Proceedings of the 13th International Annual Symposium on Computational Geometry (SCG-97)*, pages 212–220, June 1997. Extended version. Techincal report UCRL-JC-132016 Lawrence Livermore National Laboratory.